Video interpretation: human behaviour representation and on-line recognition

Van-Thinh VU, François BRÉMOND and Monique THONNAT
Project ORION of I.N.R.I.A. Sophia Antipolis,
2004 route des Lucioles, BP93-06902 Sophia Antipolis Cedex, France.
e-mail: {Thinh.Vu, Francois.Bremond, Monique.Thonnat}@sophia.inria.fr
http://www-sop.inria.fr/orion/orion-eng.html

Abstract: This paper presents a recent work in human behaviour representation and on-line recognition for video interpretation. We propose a declarative model to represent human behaviours and we use the logic-based approach to recognise pre-defined behaviour models. We demonstrate our representation formalism and the inference engine on two video sequences. We also propose a limit for the number of behaviour actors based on experimental results. The processing time is still a challenge with complex behaviour models and cluttered scenes.

Keywords: video interpretation, on-line human behaviour recognition, cognitive vision, human behaviour representation.

1. Introduction

This paper presents recent works in human behaviour representation for video understanding ([9], [10]). The goal is to recognise behaviours of humans involved in a scene depicted by a video sequence. The recognition process takes as input (1) human behaviour models predefined by experts, (2) 3D geometric information of the observed environment and (3) a video stream of persons tracked by a vision module ([2], [10]). To represent human behaviours, we first propose a language to describe behaviour models and secondly a logic-based approach to recognise in real time human behaviour occurrences.

For 20 years, the issues of temporal scenario representation and recognition have been approached. M. Ghallab [3] has represented a temporal scenario as a set of temporal constraints on time-stamped events. The recognition algorithm implements propagation of temporal constraints based on RETE algorithm. C. Pinhanez and A. Bobick [7] have implemented a simple version of Allen's interval algebra [1]. S. Hongeng and R. Nevatia [4] have proposed a behaviour recognition method by using concurrence bayesian threads to estimate the likelihood of potential scenarios. N. Rota and M. Thonnat [8] have used a declarative representation of scenarios defined as a set of spatio-temporal and logic constraints. They have reduced the constraint resolution step by checking before the consistency of the constraint network using the AC4 algorithm [6].

All these techniques allow an efficient recognition of behaviours but, they do not let the experts to describe their scenarios in a natural way. For example, most of these approaches represent an event defined at one time point but cannot manipulate events defined on intervals.

2. Human behaviour representation

Our goal is to explicit all the knowledge necessary for the system to be able to recognise human behaviours. The description of this knowledge has to be declarative and in natural terms, so that the experts of the application domain can easily define and modify it. Thus, the recognition process uses only the knowledge represented by experts through behaviour models.

We represent a human behaviour model with the list of the actors involved in the behaviour and a set of constraints on these actors ([2], [8], [10]).

An actor can be a person detected as a *mobile object* by the recognition process or a *static object* of the observed environment. A person is represented by its characteristics: his/her position in the observed environment, width, velocity,…. A static object of the environment is defined as a priori knowledge (before processing) and can be either a zone of interest (a plane polygon corresponding to an entrance zone) or a piece of equipment (a 3D object as a desk). A zone is represented by its vertices, and a piece of equipment is represented by the vertices of its bounding box. The zones and the pieces of equipment are represented in the scene context of the observed environment. Static objects and mobile objects are called *scene-object*.

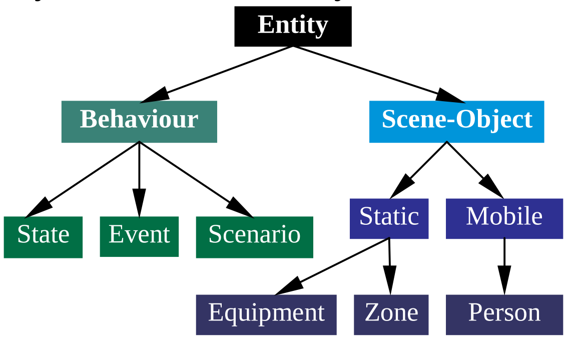

Figure 1: six types of entities are defined either as "behaviour" or as "scene-object".

In our representation, persons detected by the camera, are also characterised by relevant states, events and scenarios. A state characterises a person during a given time interval. An event defines a change of state at two successive instants. A scenario defines a combination of states, events and sub-scenarios. The notion of *behaviour* is used as a generic name for these three notions. A behaviour involves at least one person, and is defined on a time interval. An interval is represented by its starting and ending time. Defining behaviour on the time interval is important for the experts to let them describe behaviours in the more natural way. Behaviours and scene-objects are called *entities* and are defined by a generic class (see Figure 1).

To describe a human behaviour model, we have used the following definitions.

Definitions:

O - a set of scene-objects,

V - a set of variables corresponding to scene-objects,

C = {c constraint | c : $V^k \rightarrow$ BOOL, k > 0},

$\mathcal{C} = \mathcal{P}(C)$ set of parts of C,

A n-actor behaviour model is an element m = (*Actors*, *Constraints*, *Production*) of M = (V^n x $\mathcal{C}$ x $\mathcal{C}$). *Actors* is a set of n variables expressing the actors involved in the behaviour with their type and name. *Constraints* is a set of constraints defining the relationships between the actors and characterising the behaviour. *Production* is a set of deduced characteristics describing the behaviour once it has been recognised.

The n-uplet of scene-objects s = (o_1,…, o_n) is a solution at time t of the recognition process using the human behaviour model *m* if all the constraints of *Constraints* are satisfied when we assign the n variables of *Actors* with the n corresponding scene-objects of *s*. In this case the behaviour model *m* is *satisfied* at the time of the recognition of *s*. A human behaviour *b* (called instance of *m*) will be produced using *s* with the characteristics given by *production*. The human behaviour *b* is *recognised* and is added to the characteristics of the persons contained in *s*.

Figure 2 shows a model of the state "*close_to*": a person *p* is close to a piece of equipment *eq*. It involves two actors a person *p* and a piece of equipment *eq*. There is only one constraint which verifies that *p* is close to *eq*.

State(close_to,
    Actors((p : **Person**), (eq : **Equipment**))
    Constraints(
      (p ***distance*** eq <= *Close_Distance*))
    Production((s:**State**)(*Name* = "close_to")))

Figure 2: a model of the state "close_to": a person is close to a piece of equipment.

We have defined a description language to represent both parts *Constraints* and *Production* in a human behaviour model. These sets of constraints are expressed by logical predicates.

To express the logical predicates, we have used the arithmetical operators (+, -, *, /), the comparison operators (<, ≤, =, ≠, ≥, >) and the logical operators (! : not, & : and). To represent temporal relations between the behaviours, we use the operators of the interval algebra (before, after, meets,…) [1].

We use the spatial operator "distance": to calculate the distance between two scene-objects. We use also the operator "exists" to verify whether given entities exist (or not) and satisfy several constraints.

```
Event(moves_close_to,
   Actors( (p : Person), (eq : Equipment) )
   Constraints(
     (exists (state(s1, far_from, p, eq)
              state(s2, close_to, p, eq)  )
           ( (Duration of s2 ≤ 1 min)
             (s1 before s2) ) ) )
   Production((e : Event)
          ( (Interval of e := Interval of s2) ) )
```

Figure 3: model of the event "moves_close_to": a person moves close to a piece of equipment.

A model of the event "*moves_close_to*" is shown in Figure 3: a person *p* moves close to a piece of equipment *eq*. An event of this model will be recognised if p is first, far from eq and then close to eq. "**state(**s1, **far_from**, *p*, *eq*)" expresses that s1 is a state where the person p is far from the piece of equipment eq. The production part indicates how to compute the time duration of the recognised event.

On Figure 4 we show an example of a more complex scenario, "vandalism": a person p tries to "break up" a piece of equipment eq. This behaviour will be recognised if a sequence of five events described on Figure 5 has been detected.

```
Scenario(vandlism,
   Actors( (p : Person), (eq : Equipment) )
   Constraints(
     (exists(event(e1, moves_close_to, p, eq)
             event(e2, stays_at, p, eq)
             event(e3, moves_away_from, p, eq)
             event(e4, moves_close_to, p, eq)
             event(e5, stays_at, p, eq) )
         ( (e1 meets e2) (e2 meets e3)
           (e3 before e4) (e4 meets e5) ) ) )
   Production((s : Scenario)
         ( (Interval of s := Interval of e5) ) )
```

Figure 4: a model of the scenario "vandalism": a person tries to "break up" a piece of equipment.

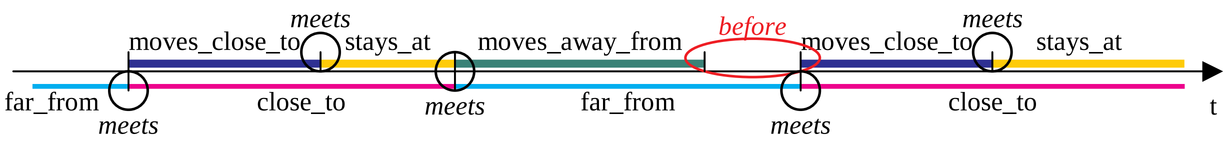

Figure 5: temporal constraints for the states and events constituting the scenario "vandalism".

3. On-line human behaviour recognition

The human behaviour recognition process has to detect from a stream of observed persons at each frame which behaviour is happening. The process takes as input (1) the behaviour models pre-defined by experts, (2) the geometric information of the observed environment and (3) the persons tracked by the vision module. We suppose that the persons are correctly tracked: their characteristics (their position in the scene, their height,…) are well detected and at two successive frames, two persons having the same name correspond to the same real person.

3.1. Selection of behaviour models

We initiate the list LM of behaviour models to be recognised with all elementary behaviour models. An elementary behaviour model is a behaviour that can be recognised at any time such as "close_to". The list LM is ordered by priority levels defined by the experts. Before the processing, for each behaviour model, we compute the set of *post-models* that corresponds to the behaviours that can be recognised once the given behaviour has been recognised. For example, once we have recognised the behaviour "close_to", it is possible that the behaviour "moves_close_to" might be recognised. So the list of *post-models* of the behaviour "close_to" contains the behaviour "moves_close_to".

Therefore, when a behaviour is recognised, its post-models (with the information about its actors) are added to the list LM of behaviour models to be recognised. The main loop to select the behaviour models at each frame is described briefly in Figure 6.

```
1: Initiate the list of models LM
     with the elementary models set
2: while LM ≠ ∅
3:   m ← get first element of LM
4:   LM ← LM - {m}
5:   Find_Solution(list of variables of m, m, ∅)
6:   if success then
         LM ← LM + post-models of m;
```

Figure 6: the main loop to select the behaviour models to be recognised.

"*Find_Solution*" (line5:) is the function that finds all the solutions (n-uplet of actors) for the model m. This function is described in the next section.

3.2. Finding solutions of a behaviour model and verification of constraint "exists"

The process to find all the solutions of a behaviour model m, is realised thanks to the function "Find_Solution". This function selects a value for each variable (actor) and check if the selected values satisfy the constraints defined within the behaviour model m. This function is described briefly in Figure 7. It takes as input the behaviour model m, the list of variables lv to be instanciated and the list of values A of the variables already instanciated. At the beginning of the process lv is initialised with all variables of m and A is the empty list. At the end of process, lv is the empty list and A contains one solution. The side effect of this function is to store the solutions once they are found.

```
1: Find_Solution(lv, m, A)
2:   v ← first variable of lv
3:   while select_value a for v in
           the domain of v [type] and verifies
           the constraints of v
4:     if v is not the last variable of lv
5:     then Find_Solution(lv - {v}, m, A + {a})
6:     else create_instance b of m with
         A+{a} in the domain of m and A [type];
```

Figure 7: finding all solutions of a behaviour model m.

To speed up the recognition process, we organise the variable domain with the variable type so the search space is reduced. There are two kinds of types: on the variable (person, equipment) and on the behaviour ("moves_close_to", "vandalism").

A second way to speed up the recognition process is to order the list of constraints for the behaviour model m. For that, we define an order on the constraints:

$$\text{order}(c) = \max_{i} \mid v_i \text{ appears in } c,$$

for example: order((v_5 before v_1)) = 5.

Then we regroup around the variable v_i every constraints that have an order i (by a pre-compilation module in the initialisation phase).

"*select_value*" (line 3:) is a process to choose a value for variable v and verify all corresponding constraints with the chosen value. A value is selected for v if it satisfies every own constraints of v. This process terminates as soon as a value is selected for v or when there is no more value to be selected.

Our method is correct because: when we choose a value for a variable v_i so every variables v_j (j < i) are already instantiated with a value, and every corresponding constraints of v_i are functions restricted to {v_1,..,v_i} so they are well calculated.

We use also this algorithm for the verification of constraint "exit", but the procedure "Find_Solution" terminates immediately when a solution is found.

3.3. Managing the recognised behaviours

Each time a behaviour b is recognised, we search a previous instance of the same behaviour in the list of recognised behaviours. If it is found and the time interval of the found entity "meets" the time interval b, then the time interval of the found entity will be prolonged. In the other case, b is stocked in the list of recognised behaviours corresponding to the behaviour model and the actors involved.

4. Results

We have modelised seventeen human behaviour models thanks to our description language. These behaviours include the models of states of a person relatively to a piece of equipment, or between two persons, models of several events that we usually meet in the scenes of metro or in an office, a model of the scenario "vandalism" and three complex models with 5, 6 and 9 actors to test the processing time. We have also tested these models on two video sequences, one (seq1) of the metro of Nuremberg (340 frames, 1 person/frame and 1 piece of equipment) and another sequence (seq2) in a FNCA bank agency (500 frames, 5 persons/frame, 20 pieces of equipment and 6 zones).

We have also measured the processing time for each frame of these two sequences. For the sequence seq1 (Figure 8), for 12 pre-defined models, the average processing time per frame is 0.19ms and the maximal processing time per frame is 0.32ms. For seq2, for 14 pre-defined models (all behaviours except the 3 complex models), the average processing time per frame is 17.5ms and the maximal processing time per frame is 34.9ms. For the same sequence, for 16 (all behaviours except the model with 9 variables), the average processing time per frame is 67.1ms and the maximal processing time per frame is 101.3ms. The model with 9 variables could not be recognised because it takes to much processing time. These tests can be found in our demo page: http://www-sop.inria.fr/orion/personnel/Thinh.Vu/demos. The experimental results show that a condition to obtain a real time processing (100ms/frame) is that the number of actors of the models cannot be superior to 6.

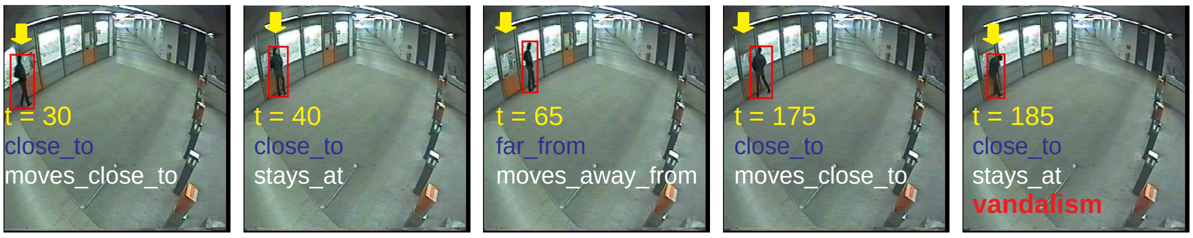

Figure 8: different instants of the scenario "vandalism" (seq1).

5. Conclusion

To represent human behaviours, we have proposed a generic model which is able to represent states, events and scenarios defined on time intervals and which is able to combine any type of constraints (logic, spatial and temporal). To help experts to describe scenario models in a declarative way, we have defined a description language. We have also proposed a recognition algorithm which can recognise in real time complex behaviours (containing up to 7 actors). We have accelerated the constraint resolution algorithm by structuring the search space of the actors.

We have validated the behaviour model and the behaviour recognition algorithm on a metro and a bank application.

The processing time of the recognition process is still a problem with complex behaviours (containing long term temporal constraints). To solve this problem we are currently trying to build special temporal operators able to check temporal relations in a bounded time.

References

[1] James F. Allen. *Towards a general theory of action and time*. Artificial Intelligence, 23:123-154, 1984.

[2] François Brémond. *Environnement de résolution de problèmes pour l'interprétation de séquences d'images*. Thèse, INRIA-Université de Nice Sophia Antipolis, 10/1997.

[3] Malik Ghallab. *On Chronicles: Representation, On-line Recognition and Learning*. 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96), Cambridge (USA), 5-8 November 1996, pp.597-606.

[4] Somboon Hongeng et Ramakant Nevatia. *Multi-Agent Event Recognition*. International Conference on Computer Vision (ICCV2001), Vancouver, B.C., Canada, 9-12/07/2001.

[5] Tony Jebara et Alex Pentland. *On Reversing Jensen's Inequality*. In Neural Information Processing Systems 13, NIPS 13, 12/2000.

[6] Roger Mohr et Thomas C. Henderson. *Arc and Path Consistency Revisited*. Research Note, Artificial Intelligence, pp225-233, vol28, 1986.

[7] Claudio Pinhanez et Aaron Bobick. *Human Action Detection Using PNF Propagation of Temporal Constraints*. M.T.T Media Laboratory Perceptual Section Technical Report No. 423, 04/1997.

[8] Nathanaël Rota et Monique Thonnat. *Activity Recognition from Video Sequences using Declarative Models*. 14th European Conference on Artificial Intelligence (ECAI 2000), Berlin, Proceeding ECAI'00 – W. Horn (ed.) IOS Press, Amsterdam, 20-25/08/2000.

[9] Cathérine Tessier. *Reconnaissance de scènes dynamiques à partir de données issues de capteurs: le projet PERCEPTION*. Rapport technique, Onera-Cert, 2 avenue Edouard-Belin, BP4025 31055 Toulouse Cedex France, 08/1997.

[10] Monique Thonnat et Nathanaël Rota. *Image understanding for visual surveillance application*. Third international workshop on cooperative distributed vision CDV-WS'99, pp51-82, Kyoto, Japan, 19-20/11/1999.